Recognition of isolated handprinted characters

Martins, Bo

Published in:
Proceedings of the IEEE International Conference on Systems, Man and Cybernetics

Link to article, DOI:
10.1109/ICSMC.1996.565504

Publication date:
1996

Document Version
Publisher's PDF, also known as Version of record

Recognition of Isolated Handprinted Characters

Bo Martins
Department of Telecommunication, Technical University of Denmark*

Abstract

Handprinted characters are of unequal complexity and a common description of all alphabet symbols seems therefore unobtainable. However, letters which are confused by human beings and by man-made OCR systems usually have approximately the same appearence and may therefore be modeled jointly. We part the set of bitmaps into types, where each type has its unique feature space. The bitmaps belonging to some type is modeled independently from bitmaps belonging to other types. The feature vector of a bitmap initially constitutes a lossy representation of the contour(s) of the bitmap. The initial feature space is usually too large but can be reduced automatically by use of a predictive code length or predictive error criterion.

1 Introduction

The nature of scripts is such that the symbols of the alphabet (letters or words) are of highly uneven complexity. This is quite natural, since the images which we associate with specific letters have been chosen to be easily distinguishable to human beings. This fact has let to the emergence of recognition systems that consist of $|\mathcal{A}|$ different models, $\{p_i(\mathbf{y}), i = 0, 1, \cdots, |\mathcal{A}| - 1\}$, where $\mathcal{A}$ denotes the alphabet and $\mathbf{y}$ is a vector of observable data (features) associated with the unknown symbol.

With these multiple, unconnected models the problem is to pick the feature space so that the rules of classification result in a low error rate. It seems appealing to pick symbol i if $p_i(\mathbf{y})$ is larger than $p_j(\mathbf{y}), j \neq i$, even though it is obviously non-optimal when the prior probabilities of different symbols are not the same and very bad when the prior probabilities are very skewed. The modelling tools being used are frequently Hidden Markov Models (HMM)[1][2] or neural networks with a fixed architecture[3]. To try out a search in architecture space for good models is impractical because the training procedure for each model is very time consuming. Even very flexible models like Hidden Markov Models and neural networks cannot fully compensate for the more or less blind choice of feature space.

To combine predictions on letter level to prediction of words the multiple models method suffers from not modelling the conditional probabilities $p(i|\mathbf{y})$, which if we had these conditional probabilities would make the classification rule on letter level and on word level trivial. The leak is patched by a substitution approach that takes advantage of the fact that recognition as, say 'n', is much less reliable than, say, 'g' on a long term basis. In many applications this approach is sufficient since the vocabulary and grammar is so restricted that the Hamming distance between valid words and/or word combinations is large.

The recognition scheme reported here is one that seeks to bring the data on such a form that enables us to build joint models for several characters, thus enabling estimation of $p(i|\mathbf{y})$, without forcing scripts of very uneven complexity to live in the same parametric world.

1.1 Previous Work

Singer and Tishby [4] [5] demonstrated that cursive scripts can be modelled extremely well when one utilizes the fundamental model of the writing proces as that of coupled oscillators of the hand muscles. The parameters of the oscillator model can be reduced to only 45 indicators, which are constant between breakpoints defined in velocity space, and 6 parameters global to the entire script. The work was directly applicable to scripts being recorded by a writing tablet. Sekita et al.[6], treated the problem of recognition of japanese characters where the timal information was unavailable. The approach, not being originated in the physiological proces of writing, defined the breakpoints as high curvature points in the bitmap of the script, the breakpoints were joined by cubic splines and breakpoints and spline parameters together formed the feature set. Takahashi and

*This work was carried out at IBM Almaden Research Center, California. I am much obliged to Drs. J. Rissanen, R. Arps, and A. Kornai for their help and good advice.

0-7803-3280-6/96/$5.00

Griffin[3] got excellent recognition results on handprinted capital letters using neural network classifiers where the features were computed on a low resolution bitmap of the original bitmap. The features were a mixture of the position of high curvature points and of frequency counts of 2 by 2 pel patterns belonging to regions of predetermined position and size. Only the 14 patterns of non-uniform color were considered and their frequency counts were collapsed into 4 counts to reduce the number of features.

2 The Raw Data

We considered preprocessed handprinted isolated characters from the NIST database. The preprocessing was done by Andras Kornai, IBM Almaden Research Center, and others, and is described in [2]. The preprocessing transforms bitmaps of uneven size to bitmaps of size $M = 24, N = 16$. The transform is an uneven subsampling which distorts the original shape (reducing legibility) and removes the absolute size information, but also to a very high degree removes noise. As shown in Fig. 1 even simple letters have highly varying appearence.

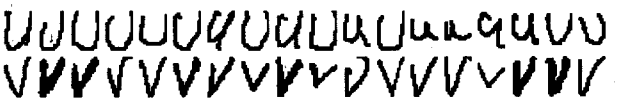

Figure 1: Examples of input data with correct classification u (top line) and v (bottom line).

3 Types and Features

The basic idea of our recognition system is to part the set of bitmaps into a number of types defined as one or more sequences of inflexion points (IP) in the bitmap. An IP-sequence describes coarsely an inner or outer contour in the bitmap and is thus a rough curve description of the script. By grouping scripts by their IP-sequences we have roughly separated the bitmaps in what may be interpreted as similar looking letters and highly skewed their prior probabilities. Moreover, our IP definition allow us to further represent the bitmaps by a detailed, type dependent description that is both compact and accurate. A classifier is built for each type.

The contour between each pair of neighbouring IPs is represented as integers which note the frequency counts of 2 by 2 pel patterns along the contour [3]. We suspect that the very reason for their success is their implicit representation of the bitmap as pieces of curves. We define an IP as a local extremum point in horizontal or vertical direction [1] (refer to Fig. 3. IPs are marked in boldface).

To connect two IPs, **A** and **B**, and to get the description of the curve connecting them we trace the contour. The procedure depends on the identity of **A**. Assume that **A** is a *Black_up*. By Fig. 2 the identity of **B** may be only a *Black_right* or a *White_down*. Stepping from a *Black_up* to a *Black_right* or a *White_down* can be done in a unique way if we define the primary direction as to-the-right and the secondary direction as down. Let the point P denote a point on the contour and let its initial position be that of the *Black_up*. Transversing from **A** to **B** we at each step register the local shape of the contour expressed as a 2 by 2 bit pattern. For a given **A** there are 4 different patterns [2] (see Fig. 3). Pick the pattern that matches the bitmap when the boxed 1 covers P, update the count for this pattern. Try to move P one step in the primary direction. If P lands on a 0 then the step is illegal and we have to choose the secondary direction. Continue like this until the next IP is reached. This happens when a boxed 1 or a boxed 0 covers **B**. If point **A** is a *White_** we first have to pick the one of the 4 patterns that matches the bitmap when the boxed 0 is placed on top of **A**, the boxed 1 is the initial position of P.

A bitmap may have more than one IP-sequence. They are found by scanning the bitmap in raster scan order for non-visited *Black_up*'s or *White_up's*.

4 Modelling Bitmaps of Equal Type

At this point an unknown bitmap belonging to type c of dimension j is represented by the pattern count vector $\mathbf{y}_c = y_{c,0} \cdots y_{c,j-1}$. We may create a proba-

[1] The choice in [6] for the IPs, although stable and low in number, cannot be used because the contour between IPs would be too complex to describe by pel pattern frequency counts. We would have to introduce other more questionable IPs or choose a different description of the contour altogether.

[2] The main advantage of our representation as compared to the original Takahashi features is the introduction of meaningful breakpoints. However, our choice of IPs gives us another advantage because the number of 2 by 2 pel patterns between adjacent IPs is usually only 4 so that we do not have to collapse frequency counts to bring down the number of features. A sufficient requirement for having only 4 different patterns between IPs is that we do not have any 2 by 2 checker board patterns in the bitmap. These possibly troublesome bitmaps may be handled by filtering out the checker board patterns, when they provoke a state of error during the process of finding and linking the IPs.

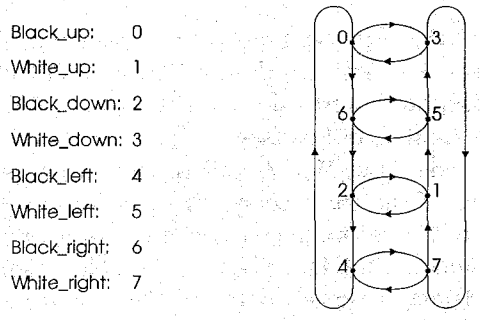

Figure 2: Valid inflexion point(IP) transitions

bilistic model $p(i|\mathbf{y}_c)$, $i \in S_c$, where S_c is the set of alphabet symbols that in the training data occurred in type c. In the following we shall omit subscript c where there can be no confusion about the type. We use the notation: $\mathbf{y} = y_0^{j-1}$.

In the training body some of the alphabet symbols will usually have been of the current type. Number these alphabet symbols $0, 1, \cdots, r$ and let their frequency counts in the training set be denoted $n_0, n_1 \cdots n_r$. To classify the current bitmap we only consider these $r+1$ possibilities. The features found during construction of the IP-sequence(s) are not necessarily well-suited for classification even though we have tried to choose features that reflect the writing process, because the dimensionality of the model should depend on j, $n_0, n_1 \cdots n_r$, and on the type. We shall consider automatic ways of reducing the feature space dimensionality so as to obtain the best possible prediction. The number of bitmaps in the training data that belong to a type is a small number (usually 20 to 200), so we must take special precautions so as not to overfit the models which must themselves be extremely simple.

4.1 The Kullback-Leibler (KL) Model

A multi-nomial model requires estimation of mean values only. If we consider $(y_0, \cdots, y_{j-1})$ as the outcome of $m = \sum_{l=0}^{j-1} y_l$ independent throws with a j-dimensional dice with the probability distribution $Q_i = (Q_{i,0}, \cdots, Q_{i,j-1})$ then by [7] we have:

$$p_i(y_0^{j-1}) \propto 2^{-mD(P||Q_i)} \tag{1}$$

Here $P = (P_0, \cdots, P_{j-1})$ denotes the normalized empirical probability distribution for the current feature vector, thus $P \cdot m = \mathbf{y}$. The Kullback-Leibler distance, D, between Q_i and P is defined as:

$$D(P||Q_i) = \sum_{l=0}^{j-1} P_l \log_2 \frac{P_l}{Q_{i,l}} \tag{2}$$

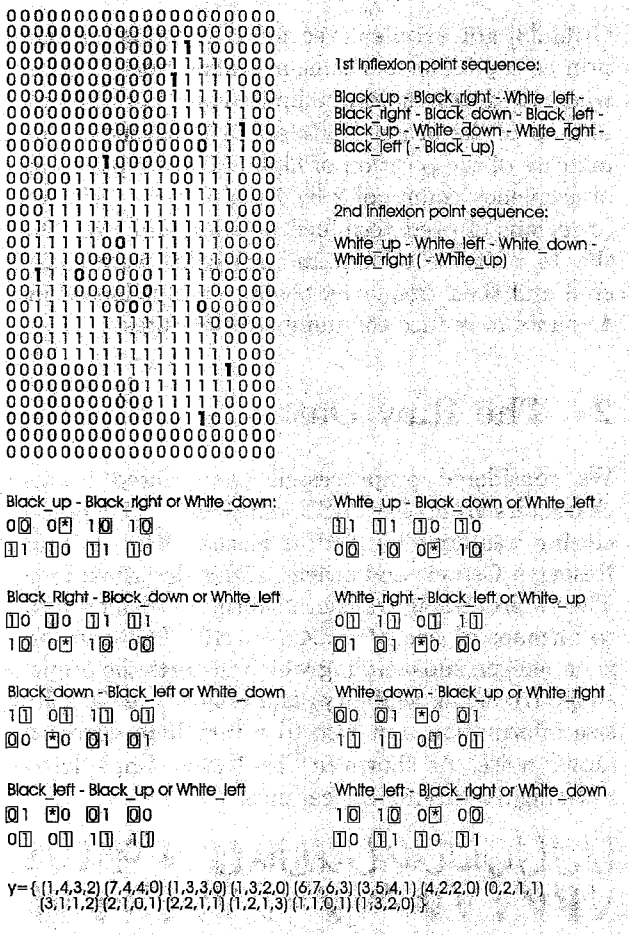

Figure 3: Construction of the contour dependent feature set. The type is $c = \{0656240374(0), 1537(1)\}$. The $10 + 4$ IP transitions give rise to a feature set, $\mathbf{y}_c$, of dimension $j = 4 \cdot (10 + 4)$.

Q_i is computed as the average value of P over the n_i instances of symbol i in the training set. When the j components of $\mathbf{y}$ are strongly dependent as certainly is the case we hope that the model will still be valid to some extent, only with a smaller 'effective value' of m. The lack of knowledge of this 'effective m' makes it impossible to exploit the fact that the prior probabilities are skewed when we have a fixed type, c. The decision rule with the Kullback-Leibler model is therefore: Pick letter i if $D(P||Q_i) = \min\{D(P||Q_0), \cdots, D(P||Q_r))\}$.

4.2 The Gaussian Models

To be able to utilize the skewedness of the prior probabilities effected by the classification into types we consider Gausian modelling:

$$p(i|y_0^{j-1}) \quad = \quad p(i|x_0^{k-1}) \tag{3}$$

$$p(i|x_0^{k-1}) = \frac{\pi_i\, p_i(x_0^{k-1})}{\pi_0\, p_0(x_0^{k-1}) + \cdots + \pi_r\, p_r(x_0^{k-1})} \quad (4)$$

$$p_i(x_0^{k-1}) = \frac{1}{(2\pi)^{k/2}\sqrt{|\Sigma_i|}} \exp(-\frac{1}{2}(\mathbf{x}' - \boldsymbol{\mu}_i')\boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)) \quad (5)$$

where $\mathbf{x} = x_0^{k-1}$ is a linear combination of the original features $\mathbf{y}$, $\mathbf{x} = \mathbf{Z}\mathbf{y}$.

Decision rule: Pick letter i if $p(i|y_0^{j-1}) = \max\{p(0|y_0^{j-1}), \cdots, p(r|y_0^{j-1})\}$.

The estimator for the prior probability of symbol i generated in type c is:

$$\pi_i = \frac{n_i + \frac{1}{2}}{n_0 + \cdots + n_r + \frac{r+1}{2}} \quad (6)$$

This estimator has unique properties[8] with respect to convergence when the distribution of the prior is multi-nomial. The mean, $\boldsymbol{\mu}_i$, and the covariance matrix, $\boldsymbol{\Sigma}_i$, are estimated by their maximum likelihood estimates over the n_i (training) bitmaps.

4.2.1 Choosing the Transform matrix

We considered two ways to pick the rows of the transform matrix. In one case (PC) we used the principal components of all the bitmaps of type c, i.e. bitmaps belonging to different letters. This method, although heavily used in classification tasks is not guaranteed to get any results and were merely used as a reference. In the other case (GG) we used a greedy search to pick out the components of the original feature vector $\mathbf{y}$ that would contribute the most information. In the pricipal components case the transform coefficients of a row are real valued, in the greedy case a row has a single 1, the rest of the coefficients are zeroes. The task is to pick the right position for the 1. We considered two greedy criteria. Assume in both cases that the $k-1$ first rows of the transform matrix have been picked already.

1. Predictive minimum description length (*pmdl*). Choose the position of the 1 so as to minimize the code length of a string stating the correct classifications of the training data given the model. Does the new row help to reduce the code length? If not we stop at the $k-1$ rows.

2. Predictive errors (*pe*). Choose the position of the 1 so that the number of errors classifying the training data is minimized. If there is more than 1 best position then choose the position that offers the lowest code length. Does the new row help to reduce the number of errors? If not we stop at the $k-1$ rows.

Type	Training: b_0^{1999}		Test: b_{2000}^{3999}	
c	$n_{u,c}$	$n_{v,c}$	$n_{u,c}$	$n_{v,c}$
06240653(0)	38	102	69	107
0624065374(0)	126	22	69	16
062403(0)	24	393	29	395
030624(0)	12	133	10	142
06240374(0)	79	34	64	34

Table 1: Binary alphabet (*uv*) - how IP-sequence grouping skew the prior probabilities. The table displays frequency counts of alphabet symbols corresponding to bitmaps with 5 most frequent types.

5 Empirical Results

5.1 Binary Alphabet

It is our observation that whenever human beings are in doubt as to the classification of an unknown character the choice stands between two candidates only - the rest of the alphabet symbols are (correctly) dismissed as possible candidates. Many man made OCR systems also have the property of being able to reduce the large number of possible alphabet symbols to 2 or 3 with a low error rate. Both cases prompt attention to the binary alphabet case. So as to better understand the properties and possible defects of our approach we first concentrated on a binary alphabet (*uv*). We considered *u*'s and *v*'s because *u*-*v* confusions is a likely source of human errors, and because *u*'s and *v*'s have usually only one contour, so that the recognition within a type becomes of significant importance. We considered a file of 4000 bitmaps, b_0^{3999}, where b_0^{999} and b_{2000}^{2999} are bitmaps of *u*'s and the rest of *v*'s.

Table 2 sheds some light on the dangers of overfitting the models. The significant difference between the recognition results on the training data and the test data when applying the Gaussian models is an indication of overfitting. The steady performance of the Kullback-Leibler model indicates that we do not overfit that model. The Gaussian models are much more vulnerable to overfitting as they require estimates of the covariance matrices, $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$. Overfitting occurs when the estimated covariance matrices have one or more small eigenvalues: we know that the resolution of the data is 1 because they are counts, hence, all healthy variance estimates should be approximately 1^2, certainly not much less. Experiments not reported here showed that the impact

Type	Training	Test	Errors		
c			KL	PC	GG
062403(0)	b_0^{1999}	b_0^{1999}	35	20	23
062403(0)	b_0^{1999}	b_{2000}^{3999}	33	18	30
062403(0)	b_{1999}^{3999}	b_{1999}^{3999}	39	22	11
062403(0)	b_0^{3999}	b_0^{3999}	73	38	36
06240653(0)	b_0^{1999}	b_0^{1999}	10	13	5
06240653(0)	b_0^{1999}	b_{2000}^{3999}	34	35	31
06240653(0)	b_{1999}^{3999}	b_{1999}^{3999}	37	30	12
06240653(0)	b_0^{3999}	b_0^{3999}	53	50	23

Table 2: Binary alphabet. Recognition of u's and v's of two frequent types in a file, b_0^{3999}, where b_0^{999} and b_{2000}^{2999} depict u's and the rest depict v's. Overfitting occurs for methods PC and GG ($\lambda_{min} = 10^{-6}$). For method PC the error rates are those that would have been obtained had we known the optimal number of principal components.

of bad estimates of the covariance matrices totally ruins the performance of the Gaussian models when applied to a larger alphabet. We patched the problem of bad covariance matrix estimates by forcing the eigenvalues to be at least λ_{min}. To do that we decomposed $\mathbf{\Sigma}_i$ into its eigenform $\mathbf{VAV}'$ changing only the diagonal matrix Λ. Experiments showed that the value of λ_{min} should be adjusted according to the greedy criterion (Table 3). Another case of overfitting occurs when we allow the transform matrix to be picked too freely. For method PC the transform coefficients constitute a large set of free parameters being estimated from very little data. The (severe) difficulty lies in determining a proper model order k. We found no solution to that problem. The greedy build-up of the transform matrix is dangerous because it is an adaptive algorithm. Experiments with the binary alphabet and a 16 letter alphabet(see below) established the greedy Gaussian algorithm as the overall best choice of the three. All our models suffer from the drawback that we cannot have more than one cluster in feature space for the same letter. A mixture density model is particularly desirable in the frequent cases where the preprocessing 'clip' some letters and leave others in their original cycloidal form. (Many bitmaps of u's suffer from this distortion.)

5.2 Bank Cheque Alphabet

To make a rough evaluation of the performance of our system as compared to established OCR systems we joined our system with one of the OCR-systems described in [2], where the alphabet was the bank cheque alphabet (*efghilnorstuvwxy*).

Each letter had appeared 1000 times in the train-

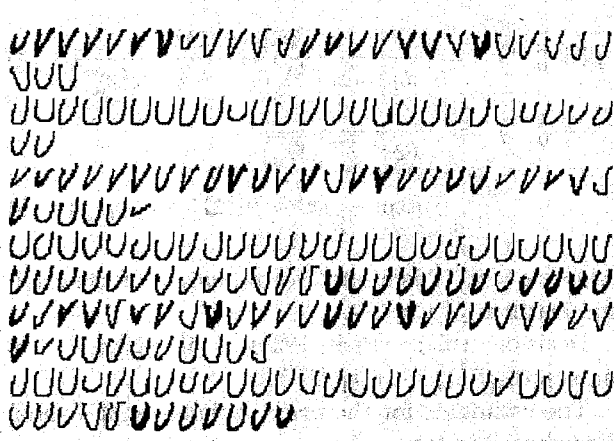

Figure 4: Recognition examples for u's and v's of type, c = 06240653(0). Rows 1–4: training body b_0^{1999}, test body b_0^{1999}. Rows 5–8: training body b_0^{1999}, test body b_{2000}^{3999}. Rows 9–12: training body b_0^{3999}, test body b_0^{3999}. Rows 1,5,9: examples of correctly recognized v's. Rows 3,7,11: examples of correctly recognized u's. Rows 2,6,10: errors where v's are taken for u's. Rows 4,8,12: errors where u's are taken for v's. Classification by the greedy Gaussian algorithm using *pmdl* to pick the transform matrix, $\lambda_{min} = 2.0$.

Type, c	Errors					
	$\lambda_{min} = 2.0$		$\lambda_{min} = 1.5$		$\lambda_{min} = 1.0$	
	pe	*pmdl*	*pe*	*pmdl*	*pe*	*pmdl*
06240653(0)	43	31	37	29	36	33
0624065374(0)	19	13	13	17	14	15
062403(0)	18	19	20	19	22	19
030624(0)	13	6	7	6	14	9
06240374(0)	14	14	14	13	13	21

Table 3: Binary alphabet (uv). Recognition results for bitmaps with the 5 most frequent IP-sequences using the greedy Gaussian algorithm.

ing set and each letter except g appeared 1000 times in the test set - there were 939 g's in the test set. The hybrid system switches between the two pure systems, letting ours do the prediction if the type of the present bitmap has occurred more than T times in the training set, T being some threshold, and the HMM do the prediction otherwise. On the larger alphabet the predictive error criterion were much better than the predictive code length criterion much so this is what we used. The results comparing the pure HMM system with the hybrid system is given in Table 4.

The preprocessing causes a lot of *i-l* confusions by filling the gap between the body and the dot of an *i*. The *y*'s are poorly recognized by the HMM for no obvious reason. The *t*'s and *f*s are badly recognized by

correct value	errors HMM	errors hybrid	correct value	errors HMM	errors hybrid
e	120	72	r	169	178
f	106	138	s	106	71
g	138	143	t	302	306
h	120	91	u	318	240
i	491	403	v	166	144
l	185	138	w	102	66
n	160	142	x	124	81
o	39	33	y	331	243

Table 4: Bank cheque alphabet (*efghilnorstuvwxy*). Recognition results for the pure HMM OCR-system and for the hybrid system using the greedy Gaussian algorithm with the predictive error criterion. $T = 80$

our mechanism because bitmaps of pen strokes that are intended to be horizontal or vertical are likely to be inflicted by unstable local extrema. The results for *r*'s and *v*'s are surprisingly good for the HMM. Visual inspection of the errors reveals that the *r*-*v* and *u*-*v* confusions that occur are forgiveable, indeed, a human being might make a few mistakes here. The most striking shortcoming of our frequency count representation is its possibly inadequate description of curves connecting far spaced IPs, e.g. the curve (cycloid) connecting the *White_down* and *Black_up* in *u*'s and *n*'s of type 03062124(0). This problem may be fixed by allowing further refinement of the curve description. The refinement should express the phase of the cycloid (see [5]). The threshold, T, may be made type dependent and its value computed as part of the training. We did not do that.

6 Conclusion

A new approach towards optical character recognition has been proposed. Results indicate that the fundamental hurdle of unequal number of strokes for different handprinted characters can be (partly) overcome, making the field of optical character recognition open to methods that build combined models for similar looking alphabet symbols. The basic idea is to part the set of bitmaps into a number of types defined as set of sequences of connected inflexion points in the bitmap. The contour between each pair of neighbouring inflexion points is represented as 4 integers which note the frequency counts of 2 by 2 pel patterns along the contour. Bitmaps of different inflexion point sequences (types) are treated as independent. To classify bitmaps of a specific type a model for the frequent alphabet symbols of that type is built in a subspace of the space of pattern counters. A greedy algorithm may be used to pick the subspace using a predictive code length or a predictive error criterion. Different, simple model classes were investigated. Fitting Gaussians in the pattern counter subspace found by the greedy algorithm gave the overall best results both for a binary alphabet case (*uv*) and for the bank cheque alphabet. Our system was combined with a traditional HMM-based OCR-system so that our system would do the prediction when the type of the present bitmap had occurred frequently, and the HMM would do the prediction otherwise. The hybrid system was somewhat better than the pure HMM-system reducing the error rate by 16 per cent. Using the original NIST data instead of the preprocessed data would probably improve the results.

References

[1] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[2] A. Kornai, K.M. Mohiuddin, and S.D. Connell, "An HMM-Based Legal Amount Field OCR System for Checks," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, 1995.

[3] H. Takahashi and T. Griffin, "Recognition Enhancement By Linear Tournament Verification," in *Proc. of the second international conference on document analysis and recognition*, Tsukaba Science City, Japan, Oct 20-22 1993.

[4] Y. Singer and N. Tishby, "Decoding Cursive Scripts," in *NIPS 6*, 1994, vol. 6.

[5] Y. Singer and N. Tishby, "Dynamical encoding of cursive handwriting," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1993.

[6] I. Sekita, K. Toraichi, R. Mori, K. Yamamoto, and H. Yamada, "Feature Extraction of Handwritten Japanese Characters by Spline Functions for Relaxation Matching," *Pattern Recognition*, vol. 21, no. 1, 1988.

[7] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, Wiley-Interscience, New York, 1991.

[8] J. Rissanen, "Fisher Information and Stochastic Complexity," *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 40–47, Jan. 1996.